

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-183750

(43)Date of publication of application : 28.06.2002

(51)Int.Cl. G06T 15/00
G06F 9/38
G06T 1/20

(21)Application number : 2001-305961

(71)Applicant : TERARIKON INC

(22)Date of filing : 02.10.2001

(72)Inventor : YEAGER MARK O
CORRELL KENNETH W
KERZNER STEVEN
LAUER HUGH C
SEILER LARRY D

(30)Priority

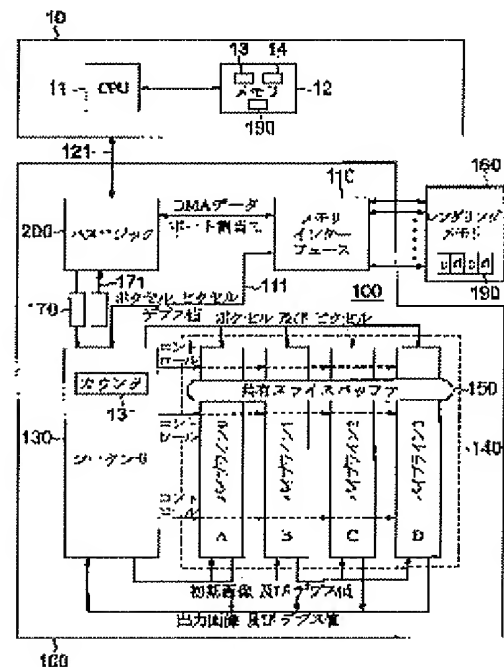
Priority number : 2000 679248 Priority date : 04.10.2000 Priority country : US

(54) RENDERING SYSTEM AND RENDERING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a rendering system capable of temporally superimposing the rendering operation over data transfer operation for a long period, and synchronizing individual operations with each other.

SOLUTION: This rendering system for executing rendering of graphic data has a host processor generating a data transfer command and a rendering command, and a memory storing the data transfer command and the rendering command in a command queue. A command parser of the system simultaneously parses and process the data transfer command and the rendering command of the command queue, and a synchronization mechanism synchronizes the processing with the parsing of the data transfer command and the rendering command, simultaneously executed.



(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号
特開2002-183750
(P2002-183750A)

(43)公開日 平成14年6月28日(2002.6.28)

(51)Int.Cl. ⁷	識別記号	F I	テーマコード [*] (参考)
G 0 6 T 15/00	2 0 0	G 0 6 T 15/00	2 0 0 5 B 0 1 3
G 0 6 F 9/38	3 7 0	G 0 6 F 9/38	3 7 0 A 5 B 0 5 7
G 0 6 T 1/20		G 0 6 T 1/20	C 5 B 0 8 0

審査請求 未請求 請求項の数16 O L (全 14 頁)

(21)出願番号	特願2001-305961(P2001-305961)	(71)出願人	599059933 テラリコン・インコーポレイテッド アメリカ合衆国 カリフォルニア州サンマ テオ市キャンパスドライブ2955スイート 325
(22)出願日	平成13年10月2日(2001.10.2)	(72)発明者	マーク・オー・イーガー アメリカ合衆国、マサチューセッツ州、ウ エスト・グラトン、オールド・オーチャー ード・ストリート 21
(31)優先権主張番号	09/679248	(74)代理人	100057874 弁理士 曾我 道照 (外6名)
(32)優先日	平成12年10月4日(2000.10.4)		
(33)優先権主張国	米国(US)		

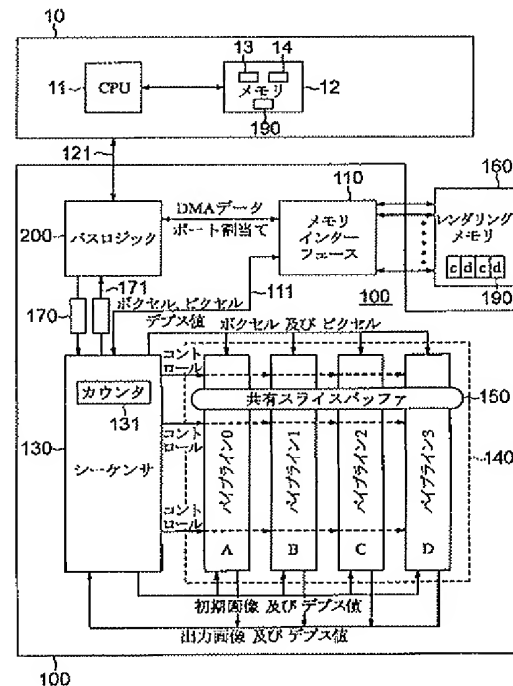
最終頁に続く

(54) 【発明の名称】 レンダリングシステム及びレンダリング方法

(57) 【要約】

【課題】 長時間にわたるレンダリング動作およびデータ転送動作を時間的に重複させることができ、かつ、個々の動作を互いに同期させることができるレンダリングシステムを得る。

【解決手段】 グラフィックデータをレンダリングするためのレンダリングシステムは、データ転送コマンドとレンダリングコマンドとを生成するホストプロセッサと、コマンドキューにデータ転送コマンドとレンダリングコマンドとを格納するメモリとを備える。そのシステムのコマンドパーサは、そのコマンドキューのデータ転送コマンドとレンダリングコマンドとを同時に構文解析し、かつ処理し、同期機構を用いて、同時に行われるデータ転送コマンドおよびレンダリングコマンドの構文解析および処理を同期させる。



【特許請求の範囲】

【請求項1】 レンダリングシステムであって、
複数のコマンドキュー内にデータ転送コマンドとレンダ
リングコマンドとを格納するメモリと、

前記複数のコマンドキューそれぞれのためのコマンドキ
ューロジックであって、各コマンドキューロジックは、
対応するコマンドキューの前記データ転送コマンドおよ
び前記レンダリングコマンドを同時にフェッチし、構文
解析し、処理する、該コマンドキューロジックと、
グラフィックデータをレンダリングしている間に、同時
に行われる前記データ転送コマンドおよび前記レンダリ
ングコマンドの前記フェッチ、前記構文解析および前記
処理を同期させるための手段とを備えるレンダリングシ
ステム。

【請求項2】 前記複数のコマンドキューに格納された
同期コマンドが前記複数のコマンドキューを同期させる
請求項1に記載のレンダリングシステム。

【請求項3】 前記同期コマンドは、前記レンダリング
システムに接続されるホストプロセッサによって生成さ
れる請求項2に記載のレンダリングシステム。

【請求項4】 前記同期コマンドは、前記レンダリング
システムのレンダリングパイプラインによって生成され
る請求項2に記載のレンダリングシステム。

【請求項5】 前記コマンドキューのうちの少なくとも
1つは、ホストメモリに格納される請求項1に記載のレ
ンダリングシステム。

【請求項6】 前記コマンドキューのうちの少なくとも
1つは、レンダリングメモリに格納される請求項1に記
載のレンダリングシステム。

【請求項7】 前記複数のキューは、前記データ転送コ
マンドと前記レンダリングコマンドとを格納するため
に、複数の要素を有する循環バッファとして構成される
請求項1に記載のレンダリングシステム。

【請求項8】 前記各コマンドキューロジックはさら
に、
前記対応するコマンドキューから前記データ転送コマン
ドと前記レンダリングコマンドとをフェッチするための
DMA状態機械と、
前記データ転送コマンドと前記レンダリングコマンドと
を構文解析し、処理し、さらに同期させるためのパース
状態機械とを備える請求項1に記載のレンダリングシ
ステム。

【請求項9】 前記データ転送コマンドはDMAインタ
ーフェースに発行され、前記レンダリングコマンドは複
数の並列パイプラインに発行される請求項1に記載のレ
ンダリングシステム。

【請求項10】 前記パース状態機械は、リセット状態
と、停止状態と、実行状態と、コマンド待ち状態と、同
期待ち状態とを有する請求項8に記載のレンダリングシ
ステム。

【請求項11】 イベントを格納するためのスクラッチ
レジスタをさらに備え、前記同期コマンドは、特定のイ
ベントの発生のために前記スクラッチレジスタを検査す
る請求項2に記載のレンダリングシステム。

【請求項12】 前記データ転送コマンドおよび前記レ
ンダリングコマンドの構文解析および処理は、前記特定
のイベントの発生を待つ間、停止される請求項11に記
載のレンダリングシステム。

【請求項13】 前記各コマンドキューは前記データ転
送コマンドと前記レンダリングコマンドとを格納するた
めの複数のエントリを含み、
第1のポインタを、フェッチされることになる次のコマ
ンドを格納する第1のエントリに格納するための、前記
各コマンドキューのためのプロデューサレジスタと、
第2のポインタを、次のコマンドを格納するための第2
のエントリに格納するための、前記各コマンドキューの
ためのコンシューマレジスタとをさらに備える請求項1
に記載のレンダリングシステム。

【請求項14】 レンダリングシステムであって、
複数のコマンドキューにデータ転送コマンドと、レンダ
リングコマンドと、同期コマンドとを格納するように構
成されるメモリと、
イベントを格納するように構成される複数のレジスタ
と、

前記複数のコマンドキューそれぞれのためのコマンドキ
ューロジックであって、各コマンドキューロジックは、
対応するコマンドキューのデータ転送コマンドとレンダ
リングコマンドとを同時にフェッチし、構文解析し、処
理し、また前記コマンドキューロジックは、データをレ
ンダリングする間に、同期コマンドを用いて前記複数の
レジスタに格納される特定のイベントのための検査を行
うことにより、同時に行われる前記データ転送コマンド
と前記レンダリングコマンドとのフェッチ、構文解析お
よび処理を同期させる、該コマンドキューロジックとを
備えるレンダリングシステム。

【請求項15】 レンダリング方法であって、
複数のコマンドキューにデータ転送コマンドとレンダリ
ングコマンドとを格納するステップと、

対応するコマンドキューのデータ転送コマンドとレンダ
リングコマンドとを同時にフェッチし、構文解析し、処
理するステップと、

グラフィックデータをレンダリングする間に、同時に行
われる前記データ転送コマンドと前記レンダリングコマ
ンドとの前記フェッチ、前記構文解析および前記処理を
同期させるステップとを含む方法。

【請求項16】 レジスタにイベントを格納するステッ
プと、

同時に行われる前記データ転送コマンドと前記レンダリ
ングコマンドとの前記フェッチ、前記構文解析および前
記処理を同期させるために、同期コマンドを用いて、特

定のイベントの発生のために前記レジスタを検査するステップとをさらに含む請求項15に記載の方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明はコンピュータグラフィックスの分野に関し、特に、並列パイプライン式のレンダリングシステムにおいてグラフィックデータをレンダリングすることに関する。

【0002】

【従来の技術】コンピュータグラフィックスアプリケーションにおいて、3次元データを視覚化する必要があるとき、多くの場合にポリウムレンダリングが用いられる。そのポリウムデータには、物理的または医学的対象物、あるいは大気、地理あるいは他の科学的モデルの走査データを用いることができ、その場合には、データの視覚化によって、データによって表される根拠をなす現実世界の構造を理解しやすくなる。

【0003】ポリウムレンダリングを用いると、物理的な物体およびモデルの内部構造ならびに外部表面の特徴が視覚化される。ボクセルは通常、ポリウムレンダリングにおいて用いられる基本的なデータ項目である。ボクセルは、その物体あるいはモデルの特定の3次元部分に関連する。各ボクセルの座標(x, y, z)は、ボクセルを、表現される物体あるいはモデル内の位置にマッピングする。

【0004】ボクセルは、その物体あるいはモデルの特定の位置に関連する1つあるいは複数の値を表す。従来技術によるポリウムレンダリングシステムでは、ボクセルによって表される値には、密度、組織タイプ、弾性あるいは速度のような多数の異なるパラメータのうちの特定の1つのパラメータを用いることができる。レンダリング中に、ボクセルは、その強度値に応じた色および不透明度(RBG α)値に変換される。その値は、視認するために2次元の画像平面上に投影することができる。

【0005】レンダリング中に頻繁に用いられる1つの技術はレイキャスティングである。その場合、1組の仮想的なレイ(視線)が、ボクセルのアレイを通して投影される。レイはいくつかの視点あるいは画像平面から生じる。ボクセルの強度値はレイに沿った点でサンプリングされ、サンプリングされる値を画素値に変換するために種々の技術が知られている。

【0006】Burgess等による1999年5月20日出願の米国特許出願第09/315,742号、タイトル「Volume rendering integrated circuit」が、参照して本明細書に援用される。そこでは、簡単な従来技術のポリウムレンダリングシステムが記載される。そのレンダリングシステムは、相互接続バスによってポリウムグラフィックスボード(VGB)に接続されるホストプロセッサを備える。そのホストプロセッサには、バス

を含む任意の種類のパーソナルコンピュータあるいはワークステーションを用いることができる。そのホストは、ホストメモリあるいはメインメモリを備える。ホストメモリには、メインメモリ、キャッシュメモリおよびディスクメモリのような、プロセッサが利用可能な内部あるいは外部の記憶装置の任意の組み合わせを用いることができる。

【0007】VGBはポリウムレンダリングチップ

(VRC)に接続されるボクセルメモリとピクセルメモリとを備える。VRCは、リアルタイムでインタラクティブなポリウムレンダリング動作に必要な全てのロジックを備える。VRCは、4つの相互に接続されるレンダリングパイプラインを含む。基本的には、VGBはホストのためのレンダリングエンジンあるいは「グラフィックスアクセラレータ」を提供する。

【0008】動作中に、ホスト内で実行されるアプリケーションソフトウェアが、レンダリングのために、ポリウムデータをVGBに転送する。詳細には、ボクセルデータは、ホストメモリからバスを介してボクセルメモリに転送される。またアプリケーションは、ボクセルメモリ内の分類テーブルのような別のデータも格納する。またアプリケーションは、パイプラインによってアクセス可能なレンダリングレジスタをロードする。これらのレジスタは、そのレンダリングが如何に実行されることになるかを指定する。全てのデータがロードされた後に、アプリケーションは、レンダリング動作を開始するためにコマンドを生成する。パイプラインはそのレンダリングコマンドを実行する。レンダリング動作が完了すると、出力装置上で表示するために、出力画像がピクセルメモリからホストあるいは3Dグラフィックスカードに移動する。Burgessのシステムでは、主な動作は逐次的に行われる。すなわち、そのシステムは、ホストからVGBにボクセルデータを書き込み、VRCにおいてレンダリングを実行し、その後、VGBからホストにピクセルデータを書き込む。

【0009】しかしながら、さらに複雑なレンダリング動作の場合、メモリ転送とレンダリング動作とを重複して行うか、あるいは以前のレンダリング動作が依然として進行している間に、1つのレンダリング動作を開始することが望ましい場合がある。さらに、効率上の理由により、ホストにおいて実行されているアプリケーションあるいは他のソフトウェアによって割り込まれたり、介入されたりすることなく、あるレンダリング動作から次のレンダリング動作まで進行することが望ましい場合がある。同様に、ソフトウェアによる割り込みあるいは介入が生じることなく、レンダリング動作の完了からデータ転送の開始まで進行することが望ましい場合がある。

【0010】たとえば、パイプラインが現在のポリウムをレンダリングしている間に、以前のポリウムのレンダリングから生成された画像をホストに転送して戻

し、レンダリングされることになる次のポリュームをロードすることができるであろう。この状況では、次のポリュームのためのレンダリング動作は、そのポリュームが完全にロードされるまで開始することを禁止され、画像のデータ転送は、その画像を生成するレンダリング動作が完了するまで開始してはならない。この状況は、データの移動が、埋め込まれた多角形構造を含むときに、急激に、さらに一層複雑になる。なぜなら、これが、レンダリングを開始する前にロードされなければならない種々のデータ源の数を増加させるためである。

【0011】

【発明が解決しようとする課題】それゆえ、本発明の目的は、長時間にわたるレンダリング動作およびデータ転送などの動作を時間的に重複させることができ、ホストソフトウェアが介入することなく、個々の動作を互いに同期させることができ、さらにホストソフトウェアがレンダリング動作の状態を判定し、進行中の動作と同期できるように、レンダリングシステムを制御するための効率的な手段を提供することである。

【0012】

【課題を解決するための手段】本発明は、データ転送コマンドおよびレンダリングコマンドを生成するホストプロセッサを備える、グラフィックスデータをレンダリングするためのレンダリングシステムを提供する。またそのシステムは、データ転送コマンドおよびレンダリングコマンドを複数のコマンドキューに格納するメモリと、複数のコマンドキューのデータ転送コマンドおよびレンダリングコマンドを同時に構文解析し、かつ処理するコマンドパーサとを備える。同期機構を用いて、データ転送コマンドおよびレンダリングコマンドを同時に構文解析し、かつ処理する動作を同期させる。

【0013】

【発明の実施の形態】パイプライン機構

図1は本発明によるポリュームレンダリングシステム全体の機構を示す。そのシステムは、バス121によってレンダリングサブシステム100に接続されるホストコンピュータ10を備える。ホストは、CPU11と、ホストメモリ12とを備える。ホストメモリ12は、グラフィカルアプリケーションソフトウェア13と、オペレーティングシステムソフトウェアと通信するグラフィックドライバ14とを格納することができる。そのソフトウェアはCPU11において実行される。またホストメモリは、ポリュームデータセット、画像、および種々のテーブルのようなレンダリングデータも格納することができる。

【0014】レンダリングエンジン100の主なモジュールは、メモリアンターフェース110、バスロジック200、シーケンサ130、および4つの並列パイプライン140である。4つのパイプラインに及ぶ一対のスライスバッファ150を除いて、パイプライン(A、

B、CおよびD)は互いに独立して動作する。好ましい実施形態では、全ての主なモジュール110、200、130、140および150は、1つのVLSIチップ上のハードウェア回路で実装される。これは、本発明を用いることができるレンダリングエンジンのタイプの一例にすぎないことに留意されたい。レンダリングエンジンには、パイプライン化された、またはパイプライン化されていない多角形レンダリングエンジン、あるいは任意の他のタイプのハードウェアで実装されたレンダリングエンジンを用いることができることにも留意されたい。

【0015】メモリアンターフェース

メモリアンターフェース110は、レンダリングメモリ160を制御する。好ましい実施形態では、レンダリングメモリ160は、8つの倍速データレート(DDR(double data rate))同期DRAMモジュールを備える。そのレンダリングメモリは、ポリューム(ボクセル)、多角形、入力および出力画像(ピクセル)、デプス値(depth values)および参照用テーブルのような、パイプライン140内のレンダリンググラフィックス対象物のために直接必要とされる全てのレンダリングデータ111のための一体化された記憶装置を提供する。以下に記載されるように、レンダリングメモリ160は、本発明によるコマンドキューバッファ190を格納することもできる。別法では、ホストCPUメモリ12がコマンドキューバッファ190を格納することができる。すなわち、任意の個々のコマンドキューバッファは、レンダリングメモリ160か、あるいはホストメモリ12かのいずれかに格納することができる。

【0016】メモリアンターフェース110は、レンダリングメモリ160への全てのメモリアクセスを実施し、バスロジック200およびシーケンサ130の要求を仲裁し、データを、サブシステム100およびレンダリングメモリ160に分散させる。この利点として、広帯域幅メモリアクセス(読出しおよび書込み)が、レンダリング動作と重複して行われる。

【0017】好ましい実施形態では、VLSI100およびレンダリングメモリ160は、PCIバス121にプラグインすることができる1枚のボードとして実装される。

【0018】シーケンサ

シーケンサ130はレンダリングエンジンを制御する。それは、どのデータがメモリからフェッチされるかを判定し、そのデータを4つのパイプライン140にディスパッチし、補間の重みのようなコントロール情報を正確な時間に個々のパイプラインに送信し、レンダリング動作からの出力画像を受信する。シーケンサそのものは、多数のレジスタによって制御される1組の有限状態機械である。これらは典型的には、特定のコマンドキューのロードレジスタコマンドに回答してバスロジック200

によって書き込まれるが、バスロジックへのP C Iアクセスを介して、ホストシステム上のソフトウェアによって直接書き込まれる場合もある。いずれの場合でも、バスロジック200がF I F O 170を介してレジスタ値をシーケンサ130に書き込む。これにより、バスロジック200の動作は、シーケンサ130の動作と切り離すことができるようになり、バスロジックおよびシーケンサが異なるサイクルタイムで動作できるようになる。

【0019】内部では、シーケンサ130は、ある時点でサンプル空間を介して1つのセクションを進め、サンプル座標を順序変更されたボクセル座標に変換し、さらに4つのパイプラインのステージによって必要とされるコントロール情報を生成するために必要とされるカウンタ131を保持する。またシーケンサは、バスロジックに、レンダリング動作のステータス171も提供する。その後、バスロジックはレンダリングステータスをホスト10に提供する。

【0020】バスロジック

図2は、バスロジック200の内部構造を示す。バスロジック200は、コマンドキューロジック300を含み、バス121を介してホスト10へのインターフェースを、またF I F O 170を介してメモリインターフェース110およびシーケンサ130へのインターフェースを提供する。ホストがパーソナルコンピュータ（P C）あるいはワークステーションである場合には、バス121には、P C I仕様バージョン2.2に準拠する64ビット、66MHz P C Iバス121を用いることができる。

【0021】このために、バスロジックは、メモリインターフェース110を介してレンダリングメモリ160にデータ（Dma In）を転送し、またレンダリングメモリ160からデータ（Dma Out）を転送するためのダイレクトメモリアクセス（DMA）動作を制御するDMAインターフェース（Dma I f）210を含む。DMA動作は、バーストモードデータ転送である。このために、バスロジックはP C Iバスマスタとして動作する。

【0022】ターゲットインターフェース（T a r g e t I f）220は、レンダリングレジスタ230の読出しおよび書き込みを制御する。これらのアクセスは、個々のレジスタおよびメモリ内の個々の場所に対する直接的な読出しおよび／または書き込みであり、ホスト10によって、あるいはP C Iバス上のいくつかの他の装置によって開始される。

【0023】またバスロジックは、メモリコントロールおよび仲裁ロジック240も備え、そのロジックは、DMA要求、およびターゲットレジスタ読出し／書き込み要求を仲裁する。またロジック240は、メモリアクセス動作を、バスロジック200とメモリインターフェース110との間のコマンドに変換する。またバスロジック

は、レンダリング動作を制御するために、シーケンサ130に直接にレジスタ値を送信し、かつシーケンサ130から戻されるステータス171を受信する。

【0024】最後に、バスロジックは、コマンドキューロジック300の3つの同一のコピー（番号201、202および203）を含む。好ましい実施形態は、3つのコマンドキューに対応する。典型的には、以下に記載されるように、ロジック201～203は、それぞれD M A I nコマンド、レンダリングコマンドおよびD M A O u tコマンドを専用に処理する。好ましい実施形態では、コマンドキューリングバッファ190が、レンダリングメモリ160あるいはホストメモリ12に格納される。別法では、そのバッファはコマンドキューロジック300の一部にすることができる。

【0025】コマンドキューロジック

図2に示されるように、そのレンダリングシステムは、3つのコマンドキュー300、すなわちDma I n 201、レンダー（R e n d e r）202およびDma O u t 203に対応する。典型的な使用法では、Dma I nコマンドキュー201は、ホストメモリ12からレンダリングメモリ160へのデータの転送を制御する。レンダリングコマンドキュー202は、ユーザが入力したパラメータにしたがってレンダリングデータ111をレンダリングする。それは、レンダリングメモリ160内の場所間のグラフィックスデータをコピーすることを含む。Dma O u tコマンドキュー203は、レンダリングメモリ160からホストメモリ12へのデータの転送を制御する。このデータには、画像あるいは部分的に処理されたレンダリングデータを用いることができる。各コマンドキューバッファ190は、ホストメモリ180内、あるいはレンダリングメモリ160内に存在することができる。

【0026】図3は、各コマンドキュー201～203を実装するロジック300を示す。各コマンドキューに対して、コマンドキューロジック300の1つのコピーが存在する。ロジック300は、コマンドキュー状態レジスタ310、DMA状態機械320、パース状態機械330、レジスタアレイ340、ポインタロジック350、およびステータスロジック360を備える。また、各コマンドキューロジックは、共有されるステータスレジスタ600にステータス情報を報告する。そのロジックの3つの全てのコピーに対してステータスレジスタは1つしか存在しないことに留意されたい。ステータスレジスタはレンダリングレジスタ230の1つである。

【0027】状態レジスタ310は、レンダリングメモリ160あるいはホストメモリ12内の関連するコマンドキューバッファ190の場所、およびバッファ内の現在の位置を含む、関連するコマンドキューの状態を規定する。具体的には、そのレジスタは、サイズ（S i z e）レジスタ311、サブコンシューマ（s u b C o n

sumer) レジスタ312、サブベース (SubBase) レジスタ313、ベース (Base) レジスタ314、コンシューマ (Consumer) レジスタ315、プロデューサ (Producer) レジスタ316 およびスクラッチ (Scratch) レジスタ317である。

【0028】コマンドキューバッファ190に格納されるデータ値は、データ転送、レンダリング動作を制御し、転送コマンドとレンダリングコマンドとの同期を可能にする。それゆえ、バッファは、コマンド(演算子) およびデータ(オペランド)を要素cおよびdとして格納する。アプリケーションソフトウェアあるいは他の手段により、要素cおよびdは、コマンドキューバッファ190内の空きエントリに順次書き込まれる。異なるコマンドcは、そのコマンドに関連する異なる量のデータdを有する場合があります、データを全く持たない場合もある。

【0029】DMA状態機械310は、パース状態機械330が要素を読み出す、あるいは「消費する」間に、その要素を、コマンドキューバッファ190からローカル記憶装置、すなわちレジスタアレイ340に読み出す。コマンドキューバッファ190が空であるか、あるいは処理されているコマンドcが、まだ発生していないイベントに同期することを必要とする場合には、関連するプロセスが中止され、さらに多くの要素がそのキューに書き込まれるか、あるいはそのイベントが発生するまで待機する。

【0030】好ましい実施形態では、各コマンドキューバッファ190は、レンダリングメモリ160内、あるいは、ホストメモリ12のようなPCIアドレス空間内のいくつかの外部メモリ内のいずれかに線形アレイとして構成される。コマンドcおよびデータd要素は、線形アレイを通して、1ステップずつインクリメントしながら書き込みおよび読み出しが行われる。線形アレイの端部に到達したときには、開始点に循環して継続するように、そのステップ動作は循環する(ラップアラウンドする)。言い換えると、バッファ190は循環バッファとして動作する。

【0031】コマンドキュー状態レジスタ310がこのプロセスを制御する。レジスタベース (Base) 314およびサイズ (Size) 311は、関連するキューの場所および長さを記述する。Baseの最下位ビットの設定値により、そのキューは、外部メモリ12、あるいはレンダリングメモリ160のいずれかに存在する。詳細には、最下位ビットが0である場合には、そのキューはPCIアドレス空間内にあり、最下位ビットが1である場合には、そのキューはレンダリングメモリ160内にある。

【0032】サイズレジスタ311は、そのキュー内の実際のエントリの数より1だけ小さい値、すなわち2の

2乗の値を有する。その後、複雑なラップアラウンドアドレス演算を必要としないように、サイズレジスタをキューポインタを有するマスクとして用いることができる。

【0033】プロデューサ (Producer) レジスタ316およびコンシューマ (Consumer) レジスタ315は、当分野において知られている標準的なリングバッファモデルにしたがって、コマンドキューに指標付けされる。詳細には、プロデューサ316レジスタは、新しいコマンドあるいはデータ要素を書き込むことができるキュー内の最初の使用していないエントリを指示する。コンシューマ315レジスタは、その時点で処理されている要素を指示する。プロデューサレジスタおよびコンシューマレジスタはいずれもサイズレジスタによってマスクされ、それにより、下位ビットがキューを表すメモリアレイへの相対的な指標を表し、上位ビットが破棄されるようにする。このマスク処理のため、インクリメントのために簡単な算術を用いることができる。ラップアラウンドのための検査を行う必要はない。

【0034】あるコマンドのための処理が完了するとき、そのコマンドキュー内の次のエントリを指示するように、ポインタロジック350がコンシューマ315をインクリメントする。そのマスクされた値がプロデューサのマスクされた値に等しい場合には、そのキューは空であり、新しいコマンドがそのキューに書き込まれるまで、構文解析が一時的に中断される。そうでない場合には、次のコマンドcが読み出され、その処理が開始される。

【0035】以前に書き込まれたコマンドが処理されている間に、ホストプロセッサ上で実行されるアプリケーションソフトウェアは、コマンドおよびデータをキューに書き込むことができる。ソフトウェアは、プロデューサ316のマスクされた値によって示されるエントリにおいて開始するキューに、コマンドおよびデータを書き込むことによりそれを実行する。その書き込みが完了するとき、そのソフトウェアは、キューに書き込まれた要素の数だけプロデューサを自動的に更新し、プロデューサがそのキュー内の次の空のエントリを指示するようにする。

【0036】コマンドキュー内の最後の要素が処理された後、そのキューの始めに次の要素が書き込まれる。当然、そのソフトウェアは、プロデューサがコンシューマに絶対に「追いつか」ないようにしなければならない。もし追いついたなら、ソフトウェアはパース状態機械330が未だに処理していないコマンドを上書きしてしまうであろう。

【0037】レジスタサブベース (SubBase) 313およびサブコンシューマ (SubConsumer) 312は、以下に記載されるコマンドキュー「サブルーチン」ファシリティを利用可能にする。各コマンド

キューに関連するスクラッチ (Scratch) レジスタ 317 は、ソフトウェアを便利にするために設けられる。このレジスタは、一時的な値を保持することができ、以下に記載される「同期 (sync)」コマンドで、ロード、格納 (ストア)、インクリメントおよび検査されることができる。

【0038】プロデューサは、以下に記載されるようなコマンドキューのリセットに応答する場合を除いて、ソフトウェアによってのみ更新され、コマンドキューロジック 300 によっては決して更新されないことに留意されたい。コンシューマおよびサブコンシューマはコマンドキューロジックによってのみ更新され、決してソフトウェアによっては更新されない。さらに、これらのレジスタは自動的に更新される。それゆえ、これらのレジスタは、プロデューサとコンシューマとの間の比較に関する競合条件を心配することなく、任意の時点で読み出すことができる。

【0039】コマンドキューにおいて用いるための他のレジスタ図 8 は、キューを管理し、レンダリングおよび DMA イベントを同期させ、外部メモリを管理する際にソフトウェアによって用いられるための 1 組のスクラッチおよび一時的レジスタ 800 のリストである。レジスタ 800 は、スクラッチ 801、スクラッチダブル 802 およびメモリ管理 803 を含む。これらの各レジスタは、以下に記載される、コマンドロードレジスタ (load Reg) およびインクリメントレジスタ (incr Reg) を用いて、ソフトウェアあるいはハードウェアによってロード、格納 (ストア) あるいはインクリメントされることができる。さらに、そのレジスタは同期 (sync) コマンドによって検査することもできる。

【0040】これらのレジスタは、ホストプロセッサに接続されるグラフィックスボードによって生成される埋め込まれた多角形を含むピクセルおよびデプスアレイの管理および転送に対応することを特に意図している。それらは、以下に記載される、コマンドロードレジスタおよびインクリメントレジスタによって自動的にロードおよびインクリメントされる。それゆえ、これらのレジスタとデータを交換し、かつ同期するコマンドキューは、更新と検査との間の競合条件を心配する必要はない。

【0041】コマンドキューの状態

図 4 に示されるように、各コマンドキューは、コマンドを構文解析する間に、5 つの状態、すなわち、実行状態 (running) 401、コマンド待ち状態 (waiting for commands) 402、同期待ち状態 (waiting for sync) 403、停止状態 (halted) 404 あるいはリセット状態 (reset) 405 のうちの 1 つの状態にあることができる。状態遷移は、その状態間の指示されたエッジ 410 として示される。たとえば、

【数 1】

停止 || リセット (HALT || RESET)

を付された遷移は、あるサイクル中に遷移が生じること示しており、その場合には、キューコントロール (以下を参照) の停止 (halt) およびリセット (reset) ビット的一方あるいは両方が、その特定のコマンドキューに当てはまることを示す。

【0042】実行状態 401 では、コマンドキューロジック 300 が、コンシューマによって指示されるコマンドで開始して、コマンドキューからコマンドおよびデータを読み出し、次に各コマンドを構文解析し、かつ処理する。

【0043】コマンド待ち状態 402 では、コンシューマのマスクされた値がプロデューサのマスクされた値に追いついたために、コマンドの構文解析が一時的に中断されている。コマンド構文解析、すなわち実行状態 401 は、プロデューサを更新した後に自動的に再開される。その時点で、そのキューから新しいコマンドが読み出され、処理される。データ転送コマンドを処理することは、DMA あるいはレジスタ転送を生じさせることを意味し、レンダリングコマンドを処理することは、レンダリングエンジン、たとえばパイプライン 140 においてレンダリング動作を生じさせることを意味する。

【0044】同期待ち状態 403 では、ある検査を満足するために、以下に記載される同期コマンドがあるレジスタの値を待ち続けているので、コマンド構文解析は一時的に中断されている。コマンド構文解析は、同期コマンドの検査が満足されるようにそのレジスタを更新した後に自動的に再開される。

【0045】停止状態 404 では、プロデューサあるいはコンシューマの値に関係なく、コマンド構文解析が停止される。停止状態は、図 5 に関して以下に記載される、キューコントロール (Queue Control) レジスタ 500 の対応する停止ビットに書込みを行う結果として、あるいは以下に記載される、停止あるいは全停止 (halt All) ビットセットを有するコマンドを構文解析する結果として生じる。構文解析は、キューコントロールレジスタ 500 の停止ビットをクリアすることによってのみ再開することができる。

【0046】コマンドキューが停止状態 404 にあるとき、コマンドキューの全状態がそのレジスタセット 310 に含まれる。コマンドキューの隠れた状態あるいは内部状態が保持されることはない。そのキューが空でない場合には、コンシューマが、処理されることになる次のコマンドを指示する。停止ビットがクリアされるとき、コマンドが、バッファ 190 からローカルレジスタアレイ 340 内に再フェッチされる。これにより、ソフトウェアが、そのキューが停止している間にコマンドキューバッファ 190 内のキュー要素を変更でき、再開前に、そのコマンドキューロジック 300 が内部バッファ 340 にそれらの変更された要素を確実に読み込むことがで

きるようになる。

【0047】最後に、リセット状態405では、コマンドキューは、定義された初期状態にリセットされている。リセット信号を除去することにより、コマンドキューは停止状態にされる。

【0048】キューコントロールレジスタ

図5に示されるように、キューコントロールレジスタ500は、対応するコマンドキューの1つの状態を制御する。このレジスタは、各コマンドキューをリセットするためのビット、コマンドキューを停止させるためのフィールド、およびDMAコマンドを処理することができる場所を厳密に制御するためのビットを与える。そのビットは、厳密(strict)ビット501と、リセットDmaOutキュー(resetDmaOutQueue)ビット502と、リセットレンダーキュー(resetRenderQueue)ビット503と、リセットDmaInキュー(resetDmaInQueue)ビット504と、停止DmaOut(haltDmaOut)ビット505と、停止レンダー(haltRender)ビット506と、停止DmaIn(haltDmaIn)ビット507とを含む。

【0049】コマンドキューのリセット

3つのリセットビット502~504の任意のビットに「1」を書き込むことにより、対応するコマンドキューがリセット状態405に入ることができる。そのレジスタ310は全て0に設定される。コマンドキューをリセットすることにより、そのキュー内のコマンドの構文解析および処理が中断される。

【0050】コマンドキューの停止

現在のコマンドの処理が完了するとき、3つの停止ビット505~507の任意のビットに「1」を書き込むことにより、対応するコマンドキューが停止状態404に入ることができる。詳細には、対応するコマンドキューが停止状態に入る前に、長い分散-収集リストを伴うDMA転送のような、その処理に長い時間を要するコマンド、あるいはレジスタの長いリストを伴うロードレジスタ(loadReg)コマンドは、完了するまで実行されるであろう。これにより、ソフトウェアは、たとえばデバッグ中にそのキューを1つずつステップできるようになる。

【0051】そのキューが、その停止ビットがセットされる時点で待ち(waiting)状態にあるとき、そのキューは直ちに停止状態に入る。その待ち状態が、満足されていない同期コマンドに関する検査の結果であった場合には、同期コマンドが割り込まれ、停止ビットがクリアされるときに同期が再度実行されるように、コンシューマポインタが設定される。

【0052】停止ビットをクリアすることにより、そのレジスタの値によって、コマンドキューは実行あるいは待ち状態のいずれかに戻ることができる。詳細には、サ

ブベースが0でない場合には、サブコンシューマによって指示されるコマンドは、再フェッチされ、実行される。サブベースが0で、コンシューマがプロデューサと同じでない場合には、コンシューマによって指示されるコマンドは再フェッチされ、実行される。コンシューマおよびプロデューサがコマンドキュー内の同じ要素を指示する場合には、そのキューは空であると見なされ、待ち状態に入る。

【0053】コンシューマあるいはサブコンシューマによって指示されるコマンドが同期コマンドである場合には、そのコマンドは任意の他のコマンドと同様に再フェッチされ、その検査が再度評価される。その検査が満足されない場合には、そのキューは再び待ち状態に入る。

【0054】また、コマンドキューは、そのキュー内に含まれるコマンドによって停止される場合もある。要約すると、コマンドキューは4つのうちの任意の方法で停止されることができる。

【0055】第一に、アプリケーションソフトウェアが、キューコントロールレジスタ500内の停止ビットを1に設定することができる。そのコマンドキューはその時点のコマンドの終了時に停止することになるが、新しいコマンドを開始しないであろう。その時点のコマンドが同期であり、検査が満足されていない場合には、同期コマンドは中止されるであろう。以下に記載される、キューステータス内の停止ビットは、そのキューが実際に停止状態に入る時点を示す。

【0056】第二に、コマンドキューは、自らのコマンドの1つに停止ビットを設定させる場合がある。この場合には、コマンドの処理は完了し、その停止ビットはキューコントロールの対応するエントリに書き込まれ、そのコマンドキューは直ちに停止状態に入る。そのコマンドが同期コマンドである場合には、同期検査が満足されるまで、その停止ビットは書き込まれない。これは、デバッグ中にコマンドを通して1つずつステップするのに有用である。

【0057】第三に、いくつかのコマンドキューは、そのコマンドの1つにおいて全停止ビットを設定させる場合がある。その場合には、全停止ビットを含むコマンドが完了するとき、キューコントロール(QueueControl)の全ての3つの停止ビットに1が書き込まれる。その1つのコマンドキューは、全停止の代わりにそのコマンドにおいて停止ビットを設定されたかのように、直ちに停止状態に入るであろう。他の2つのキューは、ソフトウェアがその停止ビットに1を書き込んだかのような動きをする。すなわち、そのキューが停止状態に入る前に、その現在のコマンドの処理は完了するが、満足されていない検査を待つ同期コマンドは中断されるであろう。以下に記載されるキューステータス(QueueStatus)600の停止ビットは、各コマンドキューが実際に停止状態に入った時点を示す。

【0058】第四に、ソフトウェアは、キューコントロールの対応するリセットビットを設定することができる。この場合には、コマンドキューは、リセットビットがクリアされた後に停止する。

【0059】DMAコマンドの厳密 (Strict) および曖昧 (Lax) コントロールキューコントロール (QueueControl) レジスタ500の厳密 (strict) ビット501は、どのコマンドキューがどのDMAコマンドを処理することができるかを判定する。

【0060】厳密ビットが1に設定されるとき、dmaInコマンドはDmaInキューからのみ処理される場合があり、dmaOutコマンドは、DmaOutキューからのみ処理される場合がある。この規則に違反があれば、全ての3つのキューの停止ビットが直ちに1に設定され、中断が発生するようになる。違反したコマンドは処理されず、パイオレーションを含むキューのコンシューマあるいはサブコンシューマは、そのコマンドを指示したままにされる。これは、3つのキューの典型的な使用法を強制し、その使用法に違反するときの誤りを検出する。

【0061】厳密ビットが0に設定されるとき、あらゆるキューにおいて、あらゆるコマンドが構文解析され、処理されることができる。これは、たとえば、ソフトウェア開発中に、重複した処理が存在しないように、レンダリングエンジンが1つのみのコマンドキューで制御されるときに有用である。また、上記のようなdmaInコマンド、レンダーコマンドおよびdmaOutコマンドの典型的な配分以外の態様で、ソフトウェアが3つのキューの中でコマンドを割り振るときに有用である。

【0062】コマンドキューステータス

図6はキューステータス (QueueStatus) レジスタ600を示す。レジスタ600は、ステータスロジック360によって保持される。このレジスタは、コマンドキューのステータス、およびそれぞれが停止されているか否かを示す。3つの各停止ビット604~606は、対応するコマンドキューが実際に停止状態404に入っていることを示す。

【0063】各コマンドキューのステータスは、対応する2ビットステータス (status) フィールド601~603において連続して更新される。そのキューが停止されていないとき、これらは連続して更新する。そのキューが停止するとき、このステータスはラッチされ、停止された時点で、どのコマンドキューが動作中であることを示す。このステータスは、たとえば、デバッグ中にソフトウェアによって検査されることができる。

【0064】キューが停止した後、実行状態 (executing) のステータスは、停止ビットが設定された時点で、コマンドの処理が進行中であることを意味する。「プロデューサ=コンシューマのための待ち状態

(waiting because Producer=Consumer)」のステータスは、そのキューが空であるため、停止ビットが設定されたときにコマンドが処理されていなかったことを意味する。「同期検査が満足されるのを待つ状態 (waiting for sync no test to be satisfied)」のステータスは、満足されていない同期コマンドが、停止ビットを設定することによって中断されたことを意味する。

10 【0065】コマンドキューコマンド

図7はキューコマンドのためのフォーマット700を示す。コマンドは、コマンドヘッダ (演算子) と、その後が続くことができるデータ (オペランド) とを含む。コマンドの処理中の典型的な動作は、レジスタをロードあるいはストアし、同期のためにレジスタ値を検査し、DMA転送を開始することである。コマンドキューの中からレンダリング動作を開始するために、1つあるいは複数のロードレジスタ (LoadReg) コマンドによってレンダリングレジスタがロードされる。

20 【0066】各コマンドヘッダは以下のフィールドを含む。cmdフィールド701の演算子コードは処理されるコマンドの種類を示す。演算子が、各演算子コードのためのオペランド705とともに、以下のサブセクションにおいて列挙される。

【0067】中断 (interrupt) フィールド702が設定されるとき、このコマンドを処理した完了時に中断が通知される。

【0068】全停止 (haltAll) フィールド703が設定されるとき、キューコントロールレジスタ500内の3つ全ての停止ビットが、現在のコマンドの完了時に1に設定される。これにより、現在のコマンドキューは停止状態に入り、そのコンシューマあるいはサブコンシューマレジスタが、次に構文解析され、かつ処理されるコマンドを指示できるようになる。他の2つのコマンドキューは、現在のコマンドの処理が完了した後に、停止状態に入る。

【0069】停止 (halt) フィールド704が設定されるとき、現在のコマンドキューは停止状態に入り、コマンド処理が完了した直後に、キューコントロールレジスタの停止ビットが1に設定される。そのコンシューマあるいはサブコンシューマレジスタは、そのキュー内の次のコマンドを指示する。

【0070】無演算コマンド

無演算 (noop) コマンドは何も実行せず、完了するのに1クロックサイクルを要する。無演算コマンド中に、コマンドヘッダ内の割込み、停止および全停止ビットが、任意の他のコマンドと同様に構文解析される。このコマンドは、アプリケーションソフトウェアによってプレースホルダとして用いられることができるか、あるいはコマンドの2つの個別のシーケンス間で停止を強制

するために用いることができる。オペランド705は無視される。

【0071】DMAコマンド2つのDMAコマンド、すなわちdmaInおよびdmaOutは、バスロジック200の中からDMA転送を生じさせるために与えられる。各DMAコマンドは、コマンドヘッダ内にカウンタ(count)オペランド705を有し、それにそのキュー内のデータ要素の所定の数のリストが後続する。各データ要素は、ホストアドレス(hostAddress)、ホストアドレスインクリメント(hostAddressInc)と、長さ(length)とを含む。どのDMAコマンドが呼び出されるかに関係なく、ホストアドレスおよび長さによって表されるDMA転送が完了まで進められる。

【0072】DMAコマンドの処理は、キューコントロールレジスタへの停止ビットの書き込みによって中断することはできない。そのコマンドの処理は、キューが停止状態に入る前に完了する。しかしながら、DMAコマンドの処理は、そのコマンドキューがリセットされるときに中断される。この中断は、カウンタ(count)が満了したか否かのいずれの場合でも、長さバイトの1回のDMA転送の正常な終了時に生じる。

【0073】キューコントロールレジスタ500において厳密ビット501が設定される場合には、dmaInコマンドは、DmaInキューからのみ呼び出され、dmaOutコマンドは、DmaOutキューからのみ呼び出される場合がある。

【0074】同期コマンド

本発明の利点として、同期(sync)コマンドは、多数のコマンドキューを互いに、かつアプリケーションソフトウェアと同期させるための柔軟な機構を提供する。特定のコマンドキューのための同期コマンドを実行することにより、オペランド705によって規定されるレジスタが、検査値(testValue)オペランド内の値と比較されるようになる(等しい、大きい、小さい等)。比較関係は、検査(test)オペランドによって規定される。同期コマンド自体は、そのコマンドヘッダのみを含む。そのヘッダにはデータ要素は後続しない。

【0075】比較結果が正しい場合には、そのコマンドキュー内の要素の構文解析が、次のコマンドで続けられる。比較結果に誤りがある場合には、そのコマンドキューは、レジスタ(reg)によって規定されるレジスタが更新され、その比較が正しい結果になるまで、待ち状態になる。すなわち、ある後の時点で、ソフトウェアあるいはいくつかの他のコマンドキューがレジスタを更新するとき、最終的に同期コマンドの検査が満足される。その後、同期コマンドを含むコマンドキューの状態は、待ち状態から実行状態に変化し、次のコマンドが、構文解析および処理のためにフェッチされる。

【0076】同期コマンドは停止に関して専用である。コマンドキューが未処理の満足されない同期コマンド、すなわち試みられたが、その検査が未だ満足されてなく、かつそのコマンドキューのためのキューコントロール内の停止ビットが、ソフトウェアあるいはいくつかの他のコマンドキューによって1に設定される同期コマンドを有する場合には、その同期コマンドは直ちに中止される。そのコマンドキューのコンシューマあるいはサブコンシューマレジスタは、それが処理されなかったかのように、同期コマンドを指示するために設定される。その後、キューコントロール内の停止ビットがクリアされるとき、同期コマンドはコマンドバッファ190からレジスタアレイ340に再フェッチされ、再試行される。これは、アプリケーションソフトウェアに、停止状態中の同期(sync)に及ぶいくつかの予測可能な制御を与える。ソフトウェアの観点から、同期は進行するか、あるいはそれが未だ試行されていなかった場合には、それが存在していた状態に留まるかのいずれかである。比較値のような隠れた状態は、停止中のコマンドキューには保持されない。

【0077】また同期コマンドは、同様に動作できるようになる再フェッチビットも含む。詳細には、再フェッチビットが1に設定される場合には、レジスタアレイ340内のコマンドキューバッファデータは、それがその時点で停止されたかのように、フラッシュされ、再フェッチされるであろう。これにより、ハードウェアが同期検査が満足されるのを待ち続けている間に、ソフトウェアは、そのキュー内のコマンドを変更できるようになり、変更されたコマンドがレジスタアレイ340内に読み込まれることを保証する。これは、たとえば、デバッグのために、および規定された同期条件が満足された後に、正確な値でのみ満たされることができる、コマンドキュー内の次の場所に「ダミー」値が書き込まれる状況の場合に有用である。

【0078】レジスタコマンド

ストアレジスタ(storeReg)、ロードレジスタ(loadReg)およびインクリメントレジスタ(incrReg)コマンドによって、レンダリングエンジンレジスタを書き込み、読み出し、インクリメントすることができるようになる。そのレジスタのデータは、PCIアドレス空間に、あるいはPCIアドレス空間から転送することができる。ストアレジスタコマンドは、そのオペランドとしてレジスタアドレスを、さらにそれに続く、キュー内のデータ要素として64ビットPCIアドレスを規定する。ロードレジスタコマンドによって、そのオペランド内のレジスタアドレスおよびマスクによって、かつマスクにおいて設定される各ビットのためのキュー内のデータ要素によって規定されるように、可変数の連続したレジスタをロードできるようになる。各データ要素は64ビットレジスタ値を表す。最後に、イン

クリメントコマンドは、そのオペランドにおいてレジスタアドレスおよびインクリメント値を含んでおり、そのコマンドによって、そのインクリメント値が所定のレジスタに加算される。

【0079】 サブルーチンおよびリターン

サブルーチン (Subroutine) およびリターン (Return) コマンドによって、「インライン」、すなわち「コマンドサブルーチン」ではない他のキューコマンドを処理できるようになる。サブルーチンコマンドによって、コマンドサブルーチンを処理できるようになり、その後、正規の順序でキューからのコマンドを処理し続ける。サブルーチンコマンドヘッダに続くデータ要素は、メモリアドレスとして取り扱われ、そのサブルーチンを呼び出すコマンドキューのサブベースレジスタ 313 に書き込まれる。リターンアドレスはコンシューマレジスタ 315 に保持され、サブルーチンコマンドおよびそのデータ要素に続くメインコマンドキュー内の要素を指示する。サブルーチンコマンドは、コマンドキューバッファの場合のように、コマンドおよびデータを含む要素のリストとして格納される。しかしながら、サブルーチンコマンドキューバッファは循環バッファではない。それは単に、リターンコマンドによって終了される、コマンドの直線的なリストである。

【0080】 好ましい実施形態では、サブルーチンコマンド自体を除く、任意のコマンドがサブルーチンコマンドキュー内に置かれる場合がある。すなわち、サブルーチンは他のサブルーチンを呼び出さない。しかしながら、他の実施形態では、サブルーチンは、付加的なオンチップサブベース (subBase) レジスタを設けることにより、あるいはチップ外のメモリにデータを格納することにより、他のサブルーチンを呼び出すこともできる。

【0081】 サブルーチンの大部分は、典型的には DMA コマンドのために用いられる。これにより、アプリケーションソフトウェアは、アクセス情報を、ある種類の動作のためのコマンドキューバッファにおいてテンプレートから都合よく切り離しておくことができる。サブルーチンは、そのコマンドキューが停止状態になるときに、中断される場合がある。そのコマンドキューのための停止ビットがクリアされるとき、サブベースレジスタは 0 ではないことに留意されたい。次のコマンドは、メインコマンドキューからの要素からではなく、サブコンシューマによって示されるサブルーチンコマンドリスト内の要素からフェッチされる。

【0082】 リターンコマンドによって、構文解析は、コンシューマによって指示されるエントリにおいてメインコマンドキューから再開されるようになる。リターンコマンドは、サブコンシューマおよびサブベースをいずれも 0 にクリアし、その結果、停止状態の後にコマンド処理を再開する場所に関して混乱は生じない。サブルー

チンコマンドの停止ビットが設定される場合には、そのサブルーチンをコールした後、そのサブルーチンの最初のコマンドを実行する前に、コマンドキューが停止する。停止されたとき、サブベースは、サブルーチンコマンドに続くコマンドのアドレスを指示し、サブコンシューマは 0 であり、コンシューマは、サブルーチンからのリターンの後に再開するためのコマンドを指示する。リターンコマンドにおいて停止ビットが設定される場合には、そのサブルーチンからリターンした後、サブルーチンコールに続く最初のコマンドを用いて進行する前に、コマンドキューは停止する。この場合には、サブベースおよびサブコンシューマはクリアされ、コンシューマが次に処理されることになるコマンドを指示する。

【0083】

【発明の効果】 こうして、本発明は、レンダリング動作から入力および出力データ転送を分離し、データ転送とレンダリング動作との間を同期させることができるコマンドキュー構造を実現する。これは、転送と動作とに長時間を要する、すなわち多くのクロックサイクルを要するときに有利である。これらの動作を別々に取り扱うことにより、特に正確な動作のために必要とされる場合、たとえばレンダリングが完了するまで、システムメモリに画像をコピーするのを遅らせる場合を除いて、長時間を要する動作が、別の各動作を待つ必要がなくなる。これらの同期遅延は、ソフトウェアの介入を必要とせず、コマンドキューコントロールロジック内で完全に処理することができる。

【0084】 本明細書に記載されるコマンドキュー構造は、実際に、任意の種類のグラフィックレンダリングエンジン、あるいは任意の他のハードウェアアクセラレータに適用することができる。本発明は、ホスト処理、サブシステム処理、および広帯域 DMA 転送を同時に必要とする多数のレンダリング動作が存在する際に特に有利である。

【0085】 本発明は好ましい実施形態を例示しながら記載されてきたが、本発明の精神および範囲から逸脱することなく、種々の他の適用形態および変更形態を実施できることは理解されたい。それゆえ、添付の請求の範囲の目的は、全てのそのような変形形態および変更形態を、本発明の真の精神および範囲内に入るように網羅することである。

【図面の簡単な説明】

【図1】 本発明によるコマンドキューを用いるレンダリングエンジンのブロック図である。

【図2】 コマンドキューロジックを備えるバスロジックのブロック図である。

【図3】 本発明によるコマンドキューを実装する好ましい実施形態のブロック図である。

【図4】 コマンドキュー状態および状態遷移のグラフを示した説明図である。

10

20

30

40

50

【図5】 キューコントロールレジスタのテーブルを示した説明図である。

【図6】 キューステータスレジスタのテーブルを示した説明図である。

【図7】 キューコマンドヘッダのテーブルを示した説明図である。

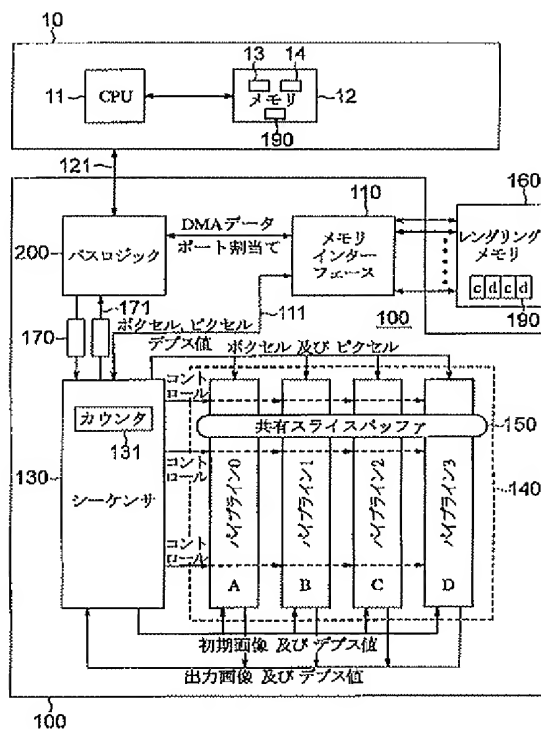
【図8】 同期のために用いられるコマンドキューレジスタのテーブルを示した説明図である。

【符号の説明】

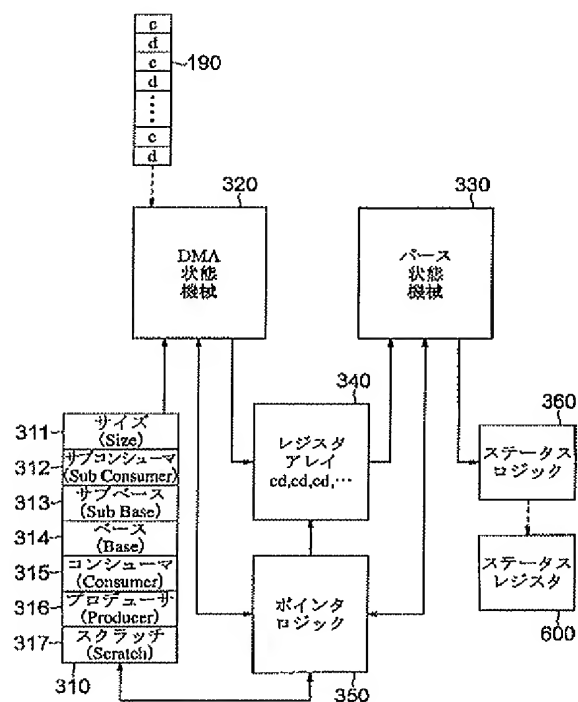
10 ホストコンピュータ、11 CPU、12 ホストメモリ、13 グラフィカルアプリケーションソフトウェア、14 グラフィックドライバ、100 レンダリングエンジン、110 メモリインターフェース、130 シーケンサ、140 並列パイプライン、150 スライスバッファ、160 レンダリングメモリ、17

0 FIFO、190 コマンドキューリングバッファ、200 バスロジック、201 DmaInコマンドキューロジック、202 レンダリングコマンドキューロジック、203 DmaOutコマンドキューロジック、210 DMAインターフェース、220 ターゲットインターフェース、230 レンダリングレジスタ、240 仲裁ロジック、300 ロジック、310 コマンドキュー状態レジスタ、320 DMA状態機械、330 パース状態機械、340 レジスタアレイ、350 ポインタロジック、360 ステータスロジック、401 実行状態、402 コマンド待ち状態、403 同期待ち状態、404 停止状態、405 リセット状態、500 キューコントロールレジスタ、600 ステータスレジスタ、700 フォーマット、800コマンドキューレジスタ。

【図1】



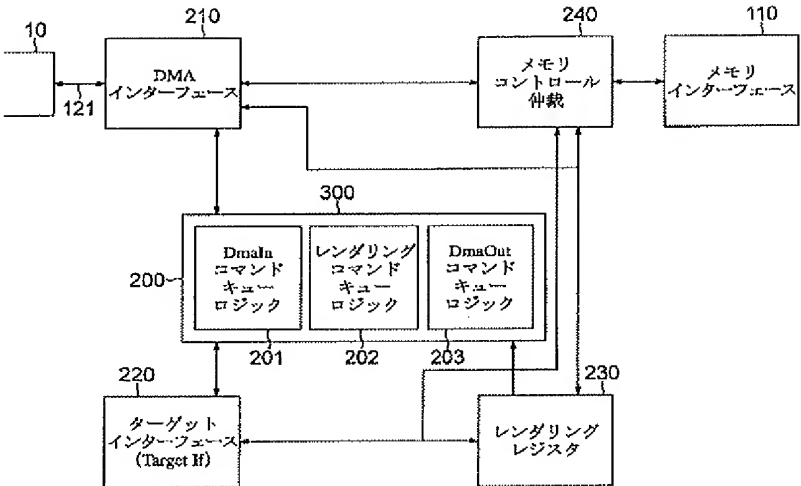
【図3】



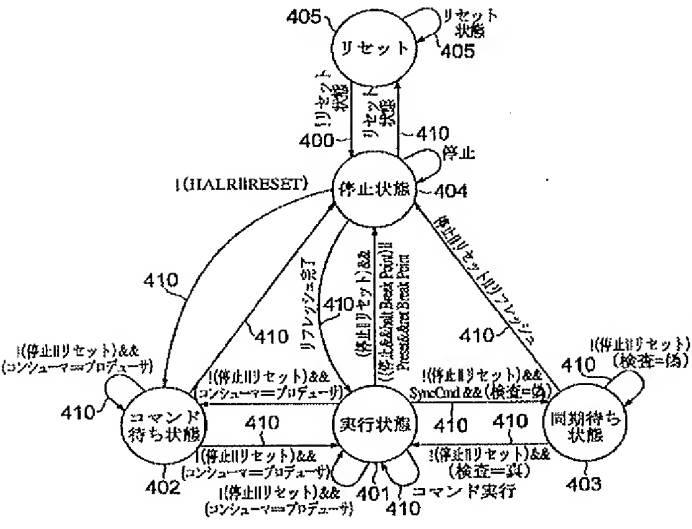
【図8】

レジスタ	タイプ	説明
801	スラッシュ	ソフトウェア使用のためのレジスタ
802	スラッシュダブル	ソフトウェア使用のための長いレジスタ
803	メモリ管理	外部のPCIメモリのどこかに存在するバッファを管理するための専用の同期レジスタ

【図2】



【図4】



【図5】

フィールド	説明
501-厳密 (Strict)	どのキューが dmaIn コマンドおよび dmaOut コマンドを実行することができるかを決定する 0: 曖昧: 任意のキューにおいて全てのコマンドが有効 1: 厳密: dmaIn コマンドは DmaIn キューにおいてのみ実行され, dmaOut コマンドは DmaOut キューにおいてのみ実行されること
502-リセット DmaOut キュー	3つのコマンドキューのためのリセットビット 0: 正常動作 1: リセット
503-リセットレンダキュー	
504-リセット DmaIn キュー	
505-停止 DmaOut	3つのコマンドキューのための停止コントロール 0: 停止条件をクリアする。メモリ (コンシューマあるいはプロデューサ) から次のコマンドをフェッチし, 実行を試みる 1: キューに現在のコマンドを停止する事を要求する
506-停止レンダ	
507-停止 DmaIn	

【図6】

フィールド名	説明
604 DmaOut停止状態	コマンドキューの停止状態: 0: そのコマンドキューは停止状態ではない 1: そのコマンドキューは停止状態、すなわち実際に停止されている
605 DmaIn停止状態	
606 レンダ停止状態	コマンドキューの状態;停止時にラッチされる
601 DmaOutステータス	
602 レンダステータス	0: 実行中 1: プロデューサ=コンシューマのための転機中 2: 同期検査が満足されるのを待機中
603 DmaInステータス	

600

【図7】

フィールド名	説明
701 コマンド(Cmd)	実行されるコマンド 8: サブルーチンコールからのリターン 7: サブルーチン 6: インクリメントレジスタ 5: ストアレジスタ 4: ロードレジスタ 3: 同期 2: dmaIn 1: dmaOut 0: 無演算
702 中断(interrupt)	このコマンドが完了したとき、中断を通知する
703 全停止(haltAll)	このコマンドが完了したとき、3つ全てのキューを停止する
704 停止(halt)	このコマンドが完了したとき、このキューを停止する
705 オペランド(operands)	特定の解釈がコマンド(cmd)に依存する

700

フロントページの続き

(72) 発明者 ケニス・ダブリュ・コレル
アメリカ合衆国、マサチューセッツ州、ランカスター、ルネンバーグ・ロード 2193

(72) 発明者 スティーヴン・カーズナー
アメリカ合衆国、マサチューセッツ州、メドフォード、アーリントン・ストリート 21、アパートメント ナンバービー

(72) 発明者 ヒュー・シー・ラウアー
アメリカ合衆国、マサチューセッツ州、コンコード、ボーダー・ロード 69

(72) 発明者 ラリー・ディー・シーラー
アメリカ合衆国、マサチューセッツ州、ボイルストン、リンデン・ストリート 198

Fターム(参考) 5B013 DD01
5B057 CH02 CH05 CH06 CH11 CH14
CH16
5B080 CA03 CA04 CA05 CA07 CA08